Task 1

The idea behind PRNG is that given a seed, it should produce a sequence that is indistinguishable from a sequence produced by a real randomness generator such that there is no polynomial time algorithm executable on a probabilistic Turing machine that can tell whether the generated sequence is random or calculated by a deterministic algorithm

$$G: \{0,1\}^k \rightarrow \{0,1\}^k$$

Hence, PRNG is an algorithm that takes a seed as an input and returns a monotonically longer output that follows the above listed property.

Build Algorithm

We first come up with a function h which on an n bit input, return n+1 bits as output. Now, this H function has to be a one-way function, as it needs to be hard to invert i.e., given a y, it is difficult to calculate x, such that H(x) = y.

To do this, we make use of dlp or discrete logarithm problem, given a prime number p and an integer x such that 0 < x < p-1 ,

$$f(x) = g^x \, mod \, p$$

Where g is a generator of the cyclic group Zp.

Now, we have a function h that on input of n bits, gives n bit output, to calculate the n+1th bit, we use Goldreich-Levin Theorem, such that n+1th bit = xi xor yi for i in range(k).

This completes our h function, which on input of n bits return n+1 bits as output.

Now, all we need to do is call the function h(x), t times, where t is some monotonically larger number than the number of bits provided as input. And we have a working PRNG.

Proof

To Prove that the above construction is a PRNG, Let

$$G(x, r) = f(x)||r||B(x, r),$$

Where x,r $\in \{0, 1\}^n$, and B(x,r) defines a hard core bit as stated in the above algorithm.

Theorem: If there exists a one-way permutation, then there exists a permutation for all polynomial stretches.

For construction, let

$$G(x) = B(f^{(m-1)}(x)) \| B(f^{(m-2)}(x)) \| \cdots \| B(f(x)) \| B(x)$$

where $f^{(i)}(x)$ represents f applied i times on x.

Input	Internal Configuration	Output
x	$f(x)$	$B(f^{(m-1)}(x))$
	$f^{(2)}(x)$	$B(f^{(m-2)}(x))$
	$f^{(3)}(x)$	$B(f^{(m-3)}(x))$
	.	.
	.	.
	.	.
	$f^{(m)}(x)$	$B(x)$

Now we can prove the same by using contradiction, such that if the above G is not a PRG, then it should not pass the a next bit text. That is, there exists an index 1 <= j <= m and an algorithm P such that,

$$\Pr[P(b_1,\ldots,b_{j-1}) = b_j \text{ for } (b_1,\ldots,b_m) = G(U_n)] > 0.5 + \epsilon$$

For some non-negligible ϵ, We construct an algorithm PRED, that can guess B(x) from f(x) with non-negligible probability. This should contradict our prediction that b is indeed hardcore.

We define PRED as follows. Given input $f(x)$:

1. Compute $f^{(2)}(x), f^{(3)}(x)), \ldots, f^{(j-1)}(x)$.
2. Compute $B(f(x)), B(f^{(2)}(x)), \ldots, B(f^{(j-1)}(x))$
3. Output $P \quad B(f^{(j-1)}(x)), \ldots, B(f^{(2)}(x)), B(f(x))$

PRED runs in polynomial time, each step consists of calculating f, B or P.

Claim: $\Pr[\text{PRED}(f(x)) = B(x)] = \Pr[P(b_1,\ldots,b_{j-1}) = b_j \text{ for } (b_1,\ldots,b_m) = G(U_n)] > \frac{1}{2} + \epsilon$.

Proof:

Since f is a permutation, it has a well-defined inverse f^{-1}.

$$G(f^{-(m-j)}(x)) = B(f^{(m-1)}(f^{-(m-j)}(x))) \| \cdots \| B(f(f^{-(m-j)}(x))) \| B(f^{-(m-j)}(x))$$

$$= B(f^{(j-1)}(x)) \| \cdots \| B(f(x)) \| B(x) \| B(f^{-1}(x)) \| \cdots \| B(f^{-(m-j)}(x)).$$

Thus as we have defined it, $\text{PRED}(f(x))$ outputs $P(b_1,\ldots,b_{j-1})$ for the first $j - 1$ bits of $(b_1,\ldots,b_m) =$

$G(f^{-(m-j)}(x))$. Moreover, we have $b_j = B(x)$. Therefore,

$$\Pr[\text{PRED}(f(x)) = B(x)] = \Pr[P(b_1,\ldots,b_{j-1}) = b_j \text{ for } (b_1,\ldots,b_m) = G(f^{-(m-j)}(U_n))]$$

But we have assumed that f is a permutation, and so $f^{-(m-j)}$ is also a permutation. And the uniformdistribution U_n is invariant under permutation. So $G(f^{-(m-j)}(U_n)) = G(U_n)$. Therefore we have

$$\Pr[\mathrm{PRED}(f(x)) = B(x)] = \Pr[P(b_1, \ldots, b_{j-1}) = b_j \text{ for } (b_1, \ldots, b_m) = G(U_n)] > 0.5 + \epsilon$$

for some non-negligible ϵ (by assumption). This proves the claim. So as argued previously, the PPT algorithm PRED predicts $B(x)$ given $f(x)$ with one half plus non-negligible probability, contradicting the fact that B is hard-core for f. Thus we conclude by contradiction that G passes all next-bit tests, and is thus a PRG.